Using a Randomized Path Planner to Generate 3D Task Demonstrations of Robot Operations

Khaled Belghith*, Froduald Kabanza* and Leo Hartman†
*Université de Sherbrooke
Sherbrooke, Québec J1K 2R1, Canada
Email: khaled.belghith@USherbrooke.ca, kabanza@USherbrooke.ca
†Canadian Space Agency
6767 route de l'Aéroport, St-Hubert Québec J3Y 8Y9, Canada
Email: Leo.Hartman@space.gc.ca

***Abstract*—In this paper we describe a new randomized path-planning approach presenting two novel features that are useful in various complex real-world applications. First, it handles zones in the robot workspace with different degrees of desirability. Given the random quality of paths that are calculated by traditional randomized approaches, this provides a mean to specify a sampling strategy that controls the search process to generate better paths by simply annotating regions in the free workspace with degrees of desirability. Second, our approach can efficiently re-compute paths in dynamic environments where obstacles and zones can change shape or move concurrently with the robot. The new path planner is implemented within an automated planning application for generating 3D tasks demonstrations involving a teleoperated robot arm on the International Space Station (ISS). A typical task demonstration involves moving the robot arm from one configuration to another. Our objective is to automatically plan the position of cameras to film the arm in a manner that conveys the best awareness of the robot trajectory to the user. For a given task, the robot trajectory is generated using the new path planner. The latter not only computes collision free paths but also takes into account the limited direct view of the ISS, the lighting conditions and other safety constraints about operating the robot. A suitable camera planning system is then used to find the best sequence of camera shots following the robot on its path.**

***Index Terms*—Probabilistic roadmaps, path planning, replanning, dynamic planning, automatic animation generation.**

I. Introduction

The Space Station Remote Manipulator (SSRMS) is a 17-meter long articulated robot arm mounted on the International Space Station (ISS). It has a complex geometry, with seven rotational joints, each with a range of ± 270. The SSRMS is a key component of the ISS and is used in the assembly, maintenance and repair of the station, and also for moving payloads from visiting shuttles. Astronauts operate the SSRMS through a workstation located inside one of the ISS compartments.

Operators manipulating the SSRMS on orbit receive support from ground operations. Part of this support consists of visualizing and validating manoeuvres before they are actually carried out. Often, the ground operators have to uplink videos to the ISS explaining how the tasks should be done. Currently these videos are programmed by computer graphics programmers based on instructions given by mission planning experts. Converging to a video that correctly presents the desired task views of the operation can take a significant amount of time.

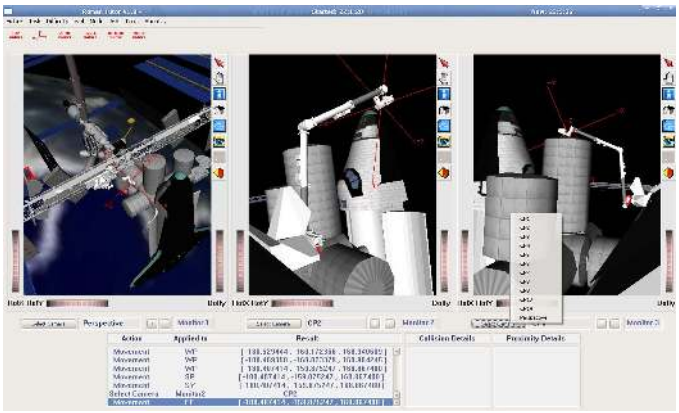


Fig. 1. Roman Tutor Interface

In order to help the ground support operations generate these video demonstrations, we have been developing an automatic task demonstration generator (ATDG) [1], which can generate 3D animations that show how to perform a given task with the SSRMS. The current ATDG prototype is integrated with a proof-of-concept simulator called ROMAN Tutor (for RObot MANipulation Tutor) for the command and control of the SSRMS that we have also developed [2]. Figure 1 shows a snapshot of the simulator. As with the physical command and control system of the ISS, the simulator has three monitors and fourteen different cameras on the exterior on the ISS. The astronaut can see the exterior only through these monitors by assigning one camera to each.

Figure 2 shows the internal architecture of the ATDG. First, ATDG calls a path planner to generate a collision free path between the two given configurations. The path is then passed to the trajectory parser which separates it into categorized segments. This will turn the continuous trajectory into a succession of scenes, where each scene can be filmed by a specific group of idioms. An idiom is a succession of shots that represents a stereotypical way to film a scene category. The parser looks for uniformity in the movements of the SSRMS to detect and recognize the category of scenes. Once the path is parsed, a call is made to the camera planner TLPlan [3] to find the best idiom that best convey each scene, while making sure

the whole sequence of shots is pleasing and comprehensive.

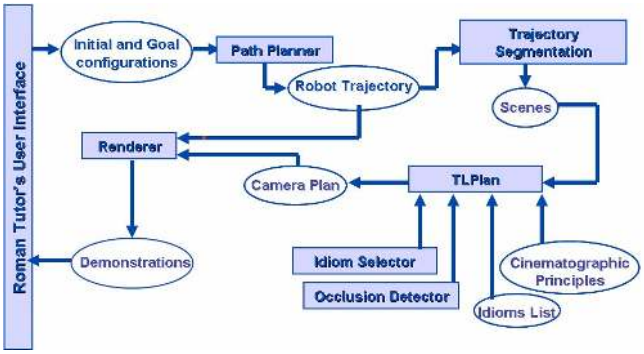


Fig. 2. ATDG architecture

Almost all tasks involve a move of the SSRMS from one configuration to another, in order for example to move a payload, or inspect a region of the ISS exterior using a camera mounted on the end effector. Computed paths must go as much as possible through cameras fields of view and avoid regions with limited visibility to enable a good appreciation of the robot motion. In other words, the cameras fields of view and the lighting conditions convey preferences for regions through which the robot path should remain or stay away from as much as possible while avoiding collisions with the ISS structure.

In order to take into account these visibility constraints on the robot path, we developed as part of ADTG a flexible adaptive path planner FAPRM that extends the probabilistic roadmap (PRM) planning framework [4], [5] to handle zones in the robot workspace with different degrees of desirability. The new path planner builds flexible roadmaps, by extending existing techniques for delay collision checking, single query and bi-direction sampling [6]. Collisions are treated as hard constraints on trajectories that must be avoided at all cost, whereas visibility constraints are handled as preferences among desirable trajectories. This allows the generation of collision-free trajectories making the robot go into regions visible through cameras and in which the manipulation is safer and easier. More specifically, we apply Genralized Adaptive A* [7] to implement a sampling strategy for fast replanning response in PRM approaches and extend it further to zones with degrees of desirability.

In the next section, we give the most relevant background. In the following section, we make a detailed study of our new flexible adaptive path planner FAPRM, fallowed with experiments that show some of its merits.

II. Background and Related Work

A configuration q of an articulated robot with n degrees of freedom (dof) is an n element vector of the robot joint positions. Since the robot moves by changing its joint rotations or translations, a path between two points is a sequence of configurations sufficiently close together connecting the two points. A path is collision-free, or in the space of collision-free configurations C_{free}, if the robot does not collide with any obstacle in the workspace in any of the configurations on the path. Computing a path is seen as making a query (to the path-planner) with the input of the goal and initial configurations. Two among today most popular approaches to path planning approaches are the combinatorial and sampling approaches.

Both combinatorial and randomized approaches discritize the problem by building an intermediate graph structure, the roadmap, and searching through it. The key difference is in what the graph represents and how it is built. With combinatorial approaches, the graph is meant to be an exact representation of C_{free} and its construction takes into account the geometry of the workspace and the robot. With sampling approaches, the graph represent samples of C_{free}. It is not an exact representation of C_{free}. The geometry of the workspace and the robot needs not be taken into account other than needed for the collision checker. Given that the configuration space is randomly sampled, randomized approaches do not guarantee a full coverage of C_{free} and they are not complete and do not guarantee optimality. In fact, they are probabilistically complete, meaning that more samples are made the closer the probability of guaranteeing the absence or presence of a solution gets to 1 [6]. Combinatorial approaches guarantee completeness and optimality, but using a sufficiently small discritization step. In practice, this results in large search spaces, making the approaches generally intractable for high dimensional configuration spaces. Sampling-based methods generally offer better performance than exact methods for domains with high-demensial spaces [8], [9]. This explains why we choose for our SSRMS application to explore a path planning solution within the PRM framework.

In the ISS environment, most of the structure is fixed, only the robots can move. However, region desirability degrees can change as well as the goal. Regions of desirability depend on the task and the involved camera views. Depending on the the orbit of the ISS, a camera may have its view towards the sun, making it undesirable. From the roadmap perspective, this means that the cost of a segment between two milestones can change dynamically. Such changes may invalidate a previously calculated path, either because it is no longer optimal, or simply because it now leads to a dead-end. Re-planning is necessary in such cases.

Dynamic path planners adapt dynamically to change happening around the robot by repairing incrementally their representation of the environment. Different approaches exist that are extensions of the A* algorithm, including D* Lite [10] and AA* [7]. Herein we adopt the approach in AA*, which was demonstrated to provide similar performance to the more famous D*, yet with a simpler defined technique. The general idea is to keep track of milestones in an optimal solution to the goal. When changes are noticed, edges costs are updated, and a new roadmap is re-computed fast, starting from the goal, taking into account previous traces of the path-calculation. Combining region preferences, and adaptive re-planning, we obtain the flexible adaptive probabilistic roadmap planner (FAPRM).

In an earlier version of our planner, we used D* for searching the roadmap [11]. This resulted in an overly complex algorithm to describe. The new version is much simpler be-

cause of the simplicity of AA* compared to AD*, yet without sacrificing performance since both AD* and AA* have similar performances [7].

III. FAPRM Planner

FAPRM implements a bias in its sampling strategy to explore the roadmap in an Adaptive A* fashion. More specifically, FAPRM is inspired from the recently introduced generalized adaptive A* path planner [7]. Adaptive A* is a heuristic search algorithm that solves similar search problems faster by updating heuristics on nodes using knowledge acquired from previous searches.

A. Algorithm Sketch

The Flexible Adaptive PRM planner (FAPRM) works with a free workspace that is segmented into zones, each zone being assigned a degree of desirability (dd), that is, a real number in the interval [0 1]. The closer is dd to 1, the more desirable the zone is. Every configuration in the roadmap is assigned a dd equal to the average of dds of zones overlapping with it. The dd of a path is an average of the dds of all configurations in the path. An optimal path is one having the highest dd.

The input for the FAPRM algorithm is thus: an initial configuration, a goal configuration, a 3D model of obstacles in the workspace, a 3D specification of zones with corresponding dd, and a 3D model of the robot. Given this input:

1) To find a path connecting the input and goal configuration, we search backward from the goal towards the initial (current) robot configuration. Backward instead of forward search is done because the robot moves, hence its current configuration is not the initial configuration, but the goal remains the same; we want to re-compute a path to the same goal but from the current position whenever the environment changes before the goal is reached.
2) A probabilistic priority queue $OPEN$ contains nodes on the frontier of the current roadmap (i.e., nodes that need to be expanded because they have no predecessor yet; or nodes that have been previously expanded but are not update anymore) and a list $CLOSED$ contains non frontier nodes (i.e., nodes already expanded)
3) Search consists in repeatedly picking a node from $OPEN$, generating its predecessors and putting the new ones and the not updated ones in $OPEN$.
   a) Every node n in $OPEN$ has a key priority proportional to the node's density and best estimate to the goal. The density of a node n, $density(n)$, reflects the density of nodes around n and is the number of nodes in the roadmap with configurations that are a short distance away. The estimate to the goal, $f(n)$, takes into account the node's dd and the Euclidean distance to the goal configuration as explained below. Nodes in $OPEN$ are selected for expansion in decreasing priority. With these definitions, a node n in $OPEN$ is selected for expansion with priority proportional to
   $$1/density(n) + f(n)$$
   b) To increase the resolution of the roadmap, a new predecessor is randomly generated within a short neighborhood radius (the radius is fixed empirically based on the density of obstacles in the workspace) and added to the list of predecessors in the roadmap generated so far; then the entire list of predecessors is returned.
   c) Collision is delayed: detection of collisions on the edges between nodes is delayed until a candidate solution is found; if colliding, we backtrack and rearrange the roadmap by eliminating nodes involved in this collision.
4) The robot may start executing the first path found.
5) Changes in the environment (moving obstacles and zones or changes in dd for zones) cause update of the roadmap and replanning.

The expression $1/density(n) + f(n)$ is separated into two components. With the first component $1/density(n)$, the selection of a node to expand works like a normal PRM [6] by selecting nodes with limited density around. With the second component $f(n)$, zone degrees of desirability and edges costs (Euclidian distance) are taken into account to seek optimality and path quality in the robot path.

The heuristic estimate is separated into two components $g(n)$ (the quality of the best path so far from n to the goal configuration) and $h(n)$ (estimate of the quality of the path from n to the initial configuration), that is, $f(n) = (g(n) + h(n))/2$; we divide by 2 to normalize $f(n)$ to values between $[0, 1]$. This definition of $f(n)$ is as in a normal A* except that:

- We do backward search, hence $g(n)$ and $h(n)$ are reversed
- The quality of a path is a combination of its dd and its cost in terms of distance traveled by the robot. Given $pathCost(n, n')$ the cost between two nodes, $g(n)$ is defined as follows:
  $$g(n) = pathdd(n_{goal}, n)/(1 + .pathCost(n_{goal}, n))$$
  with $0 \leq \gamma \leq 1$.
- The heuristic $h(n)$ is expressed in the same way as $g(n)$ and estimates the cost of the path remaining to reach n_{start}:
  $$h(n) = pathdd(n, n_{start})/(1 + \gamma.pathCost(n, n_{start}))$$

The factor γ determines the influence of the dd on $g(n)$ and on $h(n)$. With $\gamma = 0$, nodes with high dd are privileged, whereas with $\gamma = 1$ and with the dd of all nodes equal to 1, nodes with least cost to the goal are privileged. In the last case, if the cost between two nodes $pathCost(n, n'))$ is chosen to be the Euclidean distance, then we have an admissible heuristic and the algorithm guarantees converge to the optimal solution. When dds are involved, and since zones can have arbitrary configurations, it's difficult to define admissible heuristics.

B. Detailed Algorithm

Following Adaptive A* (GAA*) [12], FAPRM updates h-values with respect to the start configuration of all expanded

nodes n after every search by executing:

$$h(n) = g(n_{start}) - g(n).$$

With respect to the start configuration because the search in FAPRM is done backwards starting from the goal configuration.

This principle was first introduced in [12]. In FAPRM, we follow the generalized version of Adaptive A* (GAA*) [7]. Previous versions of Adaptive A* only allow finding shortest paths in environments where edges costs increase due to dynamic changes. GAA* addresses this issue by correcting heuristic values after edges costs decreases.

Following the same behavior in GAA*, FAPRM does not initialize all g-values and h-values up front but uses the variables *counter*, *search(n)* and *pathcost(x)* to decide when to initialize and update them by calling *UpdateState()*:

- The value of *counter* is x in the xth execution of $ComputeOrImprovePath$, that is the xth call for an Adaptive A* search on the roadmap.
- $search(n)$ stores the number of the last search that generated node n. FAPRM initializes these values to 0 for new nodes in the roadmap.
- $pathcost(x)$ stores the cost for the best path found on the roadmap by the xth search. More precisely, the formula for $pathcost(x)$ will be :

$$pathcost(x) = g(n_{start}) = pathdd(n_{goal}, n_{start})/(1 + \gamma.pathCost(n_{goal}, n_{start}))$$

Nodes in $OPEN$ are expanded in decreasing priority to update their g-values and their predecessors' g and h-values. The ordering of nodes in $OPEN$ is based on a node priority $key(n)$, which is a pair $[k_1(n), k_2(n)]$ defined as follows:

$$key(n) = [1/density(n) + f(n),\ h(n)],$$

with $f(n) = [(g(n) + h(n)]/2$ and $key(n) \leq key(n')$ if $k_1(n) \leq k_1(n')$ or ($k_1(n) = k_1(n')$ and $k_2(n) \leq k_2(n')$). During the update on nodes, FAPRM initializes the g-value of nodes not yet generated by an already performed search, nodes with $search(n) = 0$, to zero.

In the function $ComputeorImprovePath()$, when a node n with maximum key is extracted from $OPEN$, we first try to connect it to n_{start} using a fast local planner as in SBL [6].

If it succeeds, a path is then returned (Line 16). The expansion on a node n with maximum key from the $OPEN$ (Line 24) consists in sampling a new collision-free node in the neighborhood of n [6], then the sampled node takes apart in the set $Pred(n)$. After increasing the connectivity of the roadmap by adding a new node, FAPRM executes an update of the heuristics of all nodes in $Pred(n)$ in order to make them more informed and then allow for later more focused searches.

FAPRM updates the h-values of node n (line 7) if different conditions are satisfied:

- The node has not yet been generated by the current search ($search(n) \neq counter$)
- The node was generated by a previous search ($search(n) \neq 0$)

Algorithm 1 The Flexible Adaptive FAPRM Algorithm

```
01.  KEY(n)
02.    f(n) = [g(n) + h(n)]/2;
03.    return [1/density(n) + f(n); h(n)]

04.  UPDATESTATE(n)
05.    if search(n) ≠ 0
06.      if (g(n) + h(n) < pathcost(search(n))
07.        h(n) = pathcost(search(n)) − g(n);
08.        g(n) = 0;
09.    else if (search(n) = 0)
10.      g(n) = 0;
11.    search(n) = counter

12.  COMPUTEORIMPROVEPATH()
13.    while (NoPathfound)
14.      remove n with max key from OPEN;
15.      if (Connect(n, n_start))
16.        return solution path;
17.      else
18.        ExpandNode(n);;
19.        For all n' ∈ Pred(n)
20.          UPDATESTATE(n')
21.          g(n') = g(n) + c(n, n')
22.          insert n' into OPEN;
23.        insert n into CLOSED;

24.  MAIN()
25.      counter = 1;
26.      g(n_start) = g(n_goal) = 0;
27.      search(n_start) = search(n_goal) = 0;;
28.      OPEN = CLOSED = ∅
29.      UPDATESTATE(n_start)
30.      UPDATESTATE(n_goal)
31.      insert n_goal into OPEN with key(n_goal);
32.      while (Not collision-free Path)
33.          Rearrange Tree;
34.          ComputeorImprovePath();
35.          counter = counter + 1;
36.          if OPEN = ∅
37.              pathcost(search(n)) = 0
38.          else
39.              pathcost(search(n)) = g(n_start)
40.      publish current solution;
41.      while (n_start not in neighborhood of n_goal)
42.          if n_start changed
43.              if addtoTree(n_start)
44.                  publish current solution;
45.          if changes in edge costs are detected
46.              for all changed edges (u, v)
47.                  Update the edge cost c(u, v);
48.                  UpdateState(u);
49.          if significant edge cost changes observed
50.          CONSISTENCYPROCEDURE()
51.          Update the priorities for all n ∈ OPEN
52.              according to Key(n);
53.          CLOSED = ∅;
54.          while (Not collision-free Path)
55.              Rearrange Tree;
56.              ComputeorImprovePath();
57.          counter = counter + 1;
58.          if OPEN = ∅
59.              pathcost(search(n)) = 0
60.          else
61.              pathcost(search(n)) = g(n_start)
62.          publish current solution;
63.              wait for changes in edges cost;
```

- The node was expanded by the search that generated it last ($g(n) + h(n) < pathcost$ $(counter)$)

FAPRM sets $h(n)$ to the difference of the distance from n_{start} to n_{goal} during the last search that generated and expanded n, and $g(n)$ that is remained the same since the same search. FAPRM behaves like Generalized Adaptive A* (GAA*) to take into account costs increase and decrease within

the workspace. The consistency procedure (called in line 50 of the main algorithm) eagerly updates consistent h-values with respect to the start node with a version of Disjkstra's algorithm. Like GAA*, FAPRM updates the heuristics in a lazy way. However, the consistency procedure do the updates in an eager way and is run whenever costs decrease is observed. This could have a very bad impact on the performance of the planner.

IV. Experimental Results

The following experiments were run on a 1.86 GHZ Core 2 Processor with 2GB of RAM. We consider paths with a dd of 0.5 to be neutral, below 0.5 to be dangerous and above to be desirable. More specifically, dangerous zones are given a dd of 0.2 and desirable ones a dd of 0.8. A free configuration of the robot not having any contact with zones is assigned a dd of 0.5. We use $path - dd$ as a measure for path quality. We assume $\gamma = 0.7$ in all experiments. For all experiments, PRM refers to an implementation of SBL [6] for Single-query Bidirectional PRM-planner with Lazy collision detection.

We did experiments in two different scenarios. The first on a simulation of the Space Station Remote Manipulator System (SSRMS), the 17 meter long articulated robot manipulator with seven rotational joints, mounted on the International Space Station (ISS) [13]. The second on a simulated Puma robot, operating on a car. In the first scenario, we wanted to experiment and test the efficiency of the FAPRM planner to deal with very complex environments containing zones with different degrees of desirability. In the second scenario, we evaluated the "search control" capability of FAPRM.

The SSRMS on the international space station (ISS) is a very complex environment: SSRMS has 7 degrees of freedom and our ISS model contains almost 75 obstacles and 85000 triangles.

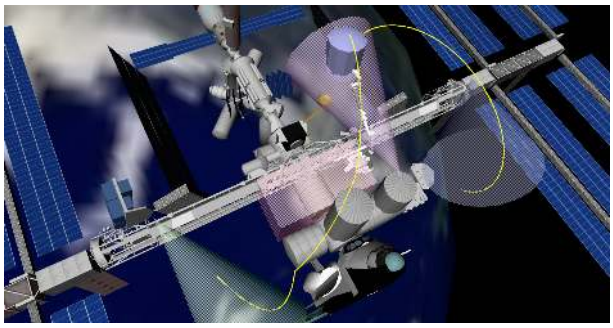


Fig. 3. SSRMS going through three different cameras fields of view (purple, green and blue cones) and avoiding a non-desired zone (red box)

Figure 3 illustrates a trajectory of the SSRMS carrying a load and going through 3 cameras fiels of view (purple, green and blue cones) and avoiding a non-desired zone with very limited lighting conditions (red box).

The first experiment illustrates the situation in which a human operator is manipulating the SSRMS from a given start configuration to a given goal configuration. Replanning occurs because the current configuration is moved arbitrary by the learner as he tries to reach the goal. The operator is not following a previously calculated path, but a path in his mind.

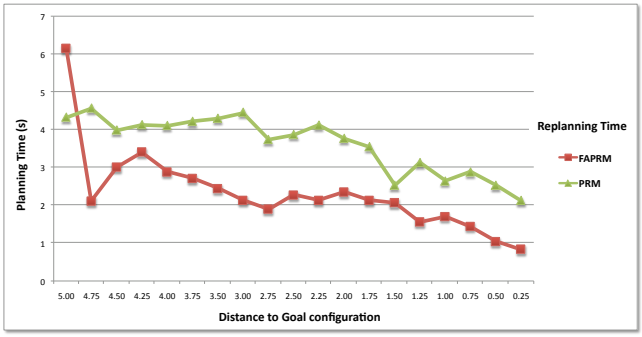


Fig. 4. FAPRM Versus PRM in Replanning

Figure 4 shows the time taken for replanning while the human operator is moving the robot toward a goal configuration in the scenario of Figure 3. We conducted the experiment two times with the operator doing exactly the same manipulations to reach the goal from the start configuration with the FAPRM and with the normal PRM. Except for the first few iterations, FAPRM take less re-planning time than PRM. For FAPRM and in the first few iterations, the overhead incurred by the GAA*-based sampling method dominates the planning time. In later iterations, it is outweighed by the savings gained by re-planning from the previous roadmap.

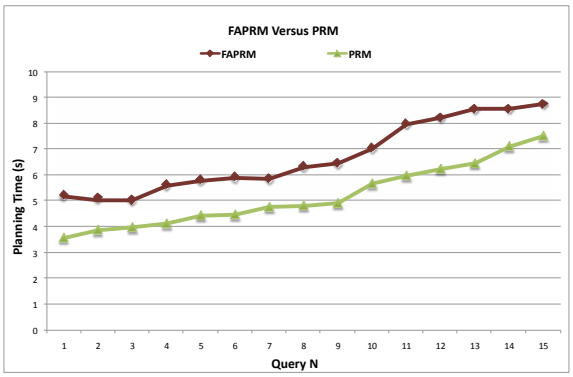


Fig. 5. FAPRM Versus PRM in Replanning

In Figure 5, we compare the time needed for FAPRM and PRM to find a solution for 15 arbitrary queries in the ISS environment. Since the time (and path quality) for finding path is a random variable given the random sampling of the free workspace, for each query we ran each of the planners 10 times and reported the average planning time (in this case, FAPRM is used in a mode not storing the roadmap between successive runs). Before displaying the results, we sorted the PRM setting in increasing order of complexity, starting with queries taking less time to solve. We see that planning with FAPRM takes more time than planning with PRM. This validates our previous analysis about FAPRM : when we seek optimality in the robot path, the time for planning will increase proportionally.

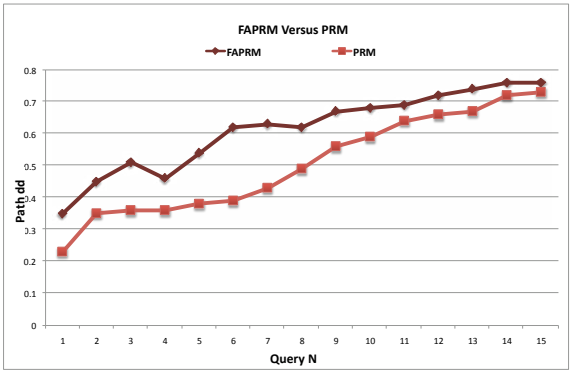
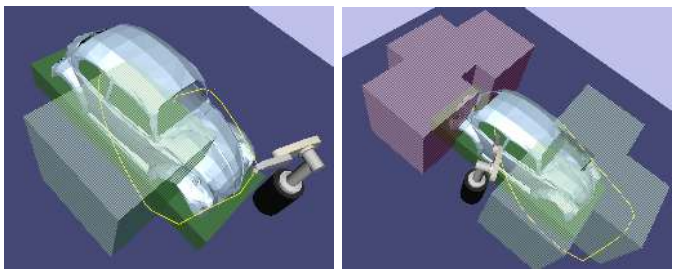


Fig. 6. FAPRM Versus PRM in Path Quality (Path-dd)

Figure 6 shows that FAPRM yields higher quality paths than PRM. This validates another previous analysis: FAPRM generates better paths than PRM, but take more time for planning. On Figure 6, PRM settings are sorted in increasing order of $path - dd$.

We also did experiments on a simulated Puma robot, operating on a car, and the results are similar to those of the SSRMS experiment. Here too the environment is very complex: 6 degrees of freedom for the robot and approximately 7000 triangles. To evaluate the "search control" capability of FAPRM, in one experiment, we specified a small desirable zone (see Fig 7), and in another, we specified a wider desirable green zone (in front of the car) and a wide undesirable red zone (in the back) (see Fig 7). In both set of experiments, we wanted to influence the sampling of the free workspace to yield paths that move the robot in front of the car (from left side, to the front, then to the right side).



(a) (b)

Fig. 7. PUMA Robot around a car

By specifying a desirable zone on the right front of the car as shown in Figure 7, and running our FAPRM many times on the same query (input/goal configuration), they yielded better paths, on average, than PRM. On the other hand, by enlarging the size and coverage of the desirable zone and adding a undesirable red zone (right, on the back of the car), as shown in Fig 7-(b), we noticed that the quality of paths increased by a percentage of 50% over 100 trials. The second experiment succeeds more often because the path is more constrained; a wider desirable zone on the front of the car together with an undesirable zone on the back of the car, make the probability of sampling a configuration along the desirable region higher than in the first set-up. Given the random quality of paths that are calculated by traditional PRM approaches, zones with degrees of desirability provide a mean to specify a sampling strategy that controls the search process to generate better paths by simply annotating the 3D workspace with region's degrees of desirability.

V. Conclusion

We just introduced a new randomized path planning approach FAPRM that implements a simple adaptive sampling strategy to bring improvements to randomized path-planning approaches, along two dimensions: (1) modeling zones in the robot workspace with different degrees of desirability, (2) efficiently re-computing paths in dynamic environments. Besides extending the range of problems solvable by randomized approaches along these dimensions, we demonstrated that even for traditional problems, our extension can improve the quality of generated paths and compensate for the random search by providing desirable regions as a mean of controlling search.

FAPRM planner is implemented within the automatic task demonstration ATDG, a system intended to improve the ground support operations on the SSMS. Although motivated by the ISS application, ATDG stays applicable for filming other kinds of complex animated scenes.

References

[1] F. Kabanza, K. Belghith, P. Bellefeuille, B. Auder, and L. Hartman, "Planning 3d task demonstrations of a teleoperated space robot arm," in *ICAPS*, 2008, pp. 164–173.

[2] F. Kabanza, R. Nkambou, and K. Belghith, "Path-planning for autonomous training on robot manipulators in space," in *International Joint Conference In Artificial Intelligence (IJCAI)*, 2005, pp. 1729–1731.

[3] F. Bacchus and F. Kabanza, "Using temporal logics to express search control knowledge for planning," *Artificial Intelligence*, vol. 116, no. 1-2, pp. 123–191, 2000.

[4] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," in *IEEE Transactions on Robotics and Automation*, vol. 12, 1996, pp. 566–580.

[5] R. Bohlin and L. Kavraki, "Path planning using lazy prm," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 521–528.

[6] G. Sanchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Ninth International Symposium on Robotics Research*, 2001, pp. 403–417.

[7] X. Sun, S. Koenig, and W. Yeoh, "Generalized adaptive a*," in *International Conference on Autonomous Agents and Multiagent Systems(AAMAS)*, 2008, pp. 469–476.

[8] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, 2005.

[9] S. M. Lavalle, *Planning Algorithms*. Cambridge University Press, 2006.

[10] S. Koenig and M. Likhachev, "D*lite," in *Proc. of the National Conference on Articial Intelligence(AAAI/IAAI)*, 2002, pp. 476–483.

[11] K. Belghith, F. Kabanza, L. Hartman, and R. Nkambou, "Anytime dynamic path-planning with flexible probabilistic roadmaps," in *ICRA*, 2006, pp. 2372–2377.

[12] S. Koenig and M. Likhachev, "Adaptive a*," in *International Conference on Autonomous Agents and Multiagent Systems(AAMAS)*, 2005, pp. 1311–1312.

[13] N. Currie and B. Peacock, "International space station robotic systems operations: A human factors perspective," in *Habitability and Human Factors Office (HHFO), NASA Johnson Space Center*, 2002.